

OS: Part 1

OS: 1) Program which manages all other programs in a computer

2) interface between hardware and application software



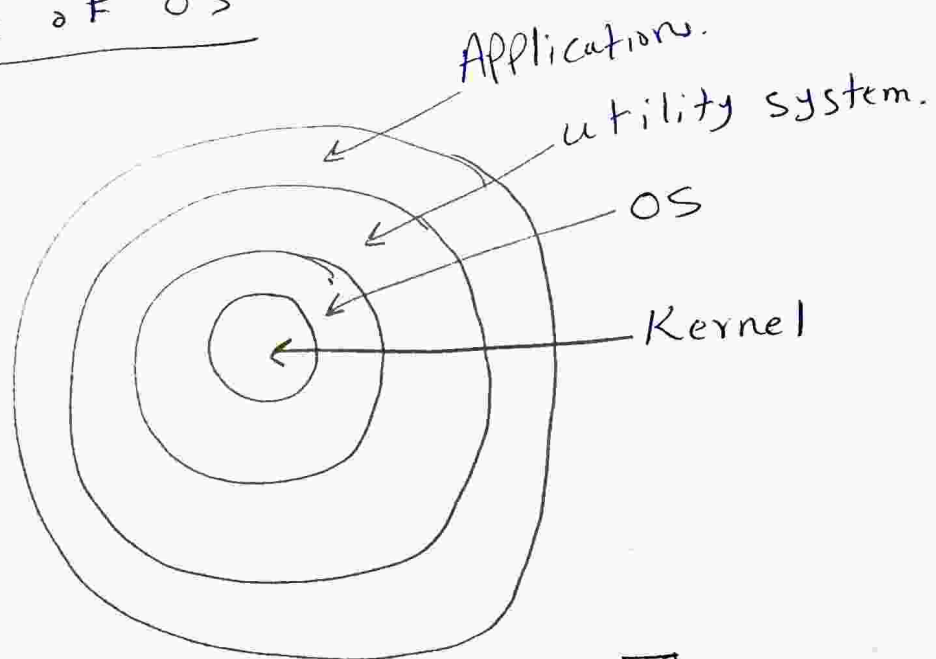
user view

- 1) varies by ~~at~~ the interface being used.
- 2) sys. consists of monitor, Keyboard ~~and~~ mouse ~~and~~

system views

- 1) we can view an OS as resource allocator.
- 2) OS is program that is most intimate with hardware.

structure of OS



* Kernel

- 1) the core of OS.
- 2) called real time executive.
- 3) Contains all the devices that interact with the hardware.

* OS

- controls and coordinates the use of hardware among the various applications programs for various users.

* Utility system

- system utilities are program that perform individual, specialized management tasks.

* Applications

- such as word processor, spreadsheet.
- compilers define the way in which these resources are used

Types of operating system

1) single user

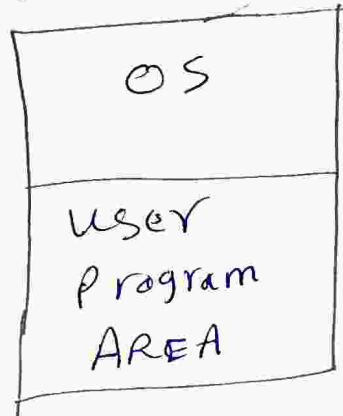
- only one user can operate on single "OS" such as "MS-DOS"

2) Multi user

- users can operate of single operating system.
- ~~also~~ allow many different users to take advantage of computer resources. for example "Windows"

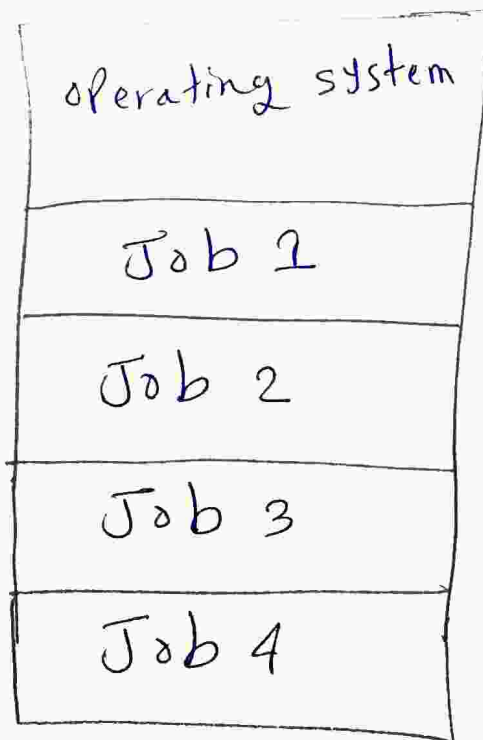
3) Batch system

- user can execute programs in batch form.
- each batch contains similar type of programs.
- batch is made by server & executed by OS.



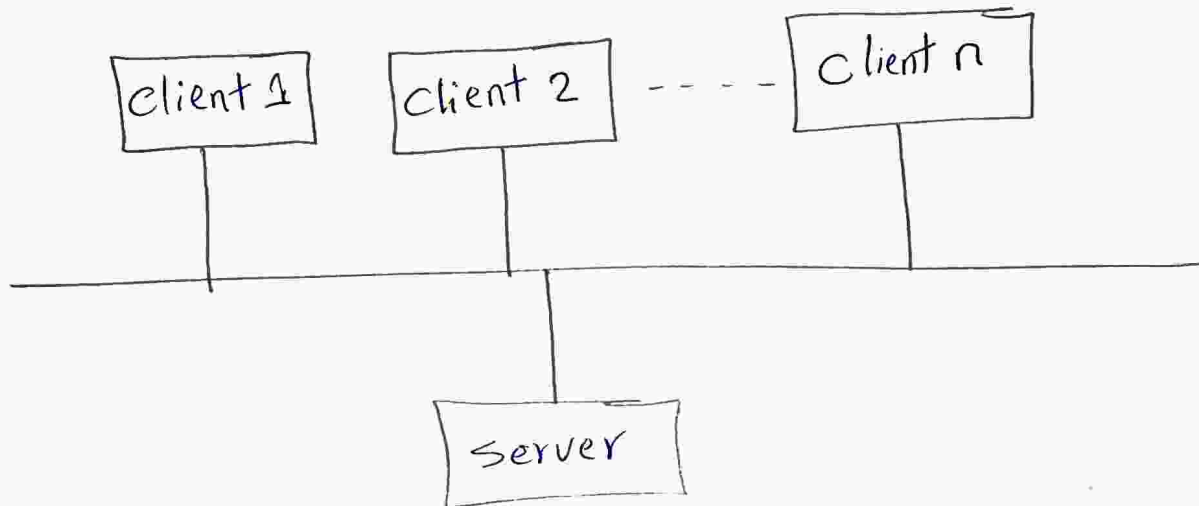
4) Multiprogramming system

- OS keeps several programs in memory at the same time.
- increase CPU utilization by organizing job so that CPU always has one to execute



5) Peer to Peer Systems

→ Systems are connected with straight line and first system will be considered as server and other systems are known as clients.



6) Distributed system

→ It is similar to client server architecture.
→ clients request the server for their requirements and server would provide all server ^{ices} that is needed to clients.
→ allow multiple app. program to cooperate to solve participate task.

7) Real time :-

→ Consists the time which is requested to the execution of programs.

→ two types (hard time - soft time)

Services to operating system

* Program execution

- system must be able to load a program into memory and to run that program.
- Program must be able to end its execution normally or abnormally.

* I/O operations

- running program may be required I/O.
- OS must provide a means to do.

* File system manipulation

- the file system is of need to read and write file.
- Program also need to create and delete file by name.

* Communications

- It may be implemented via shared memory or by the technique of message passing.

* Error detection

- OS needs to be aware of possible error.
- Error may be occur in CPU and memory hardware.

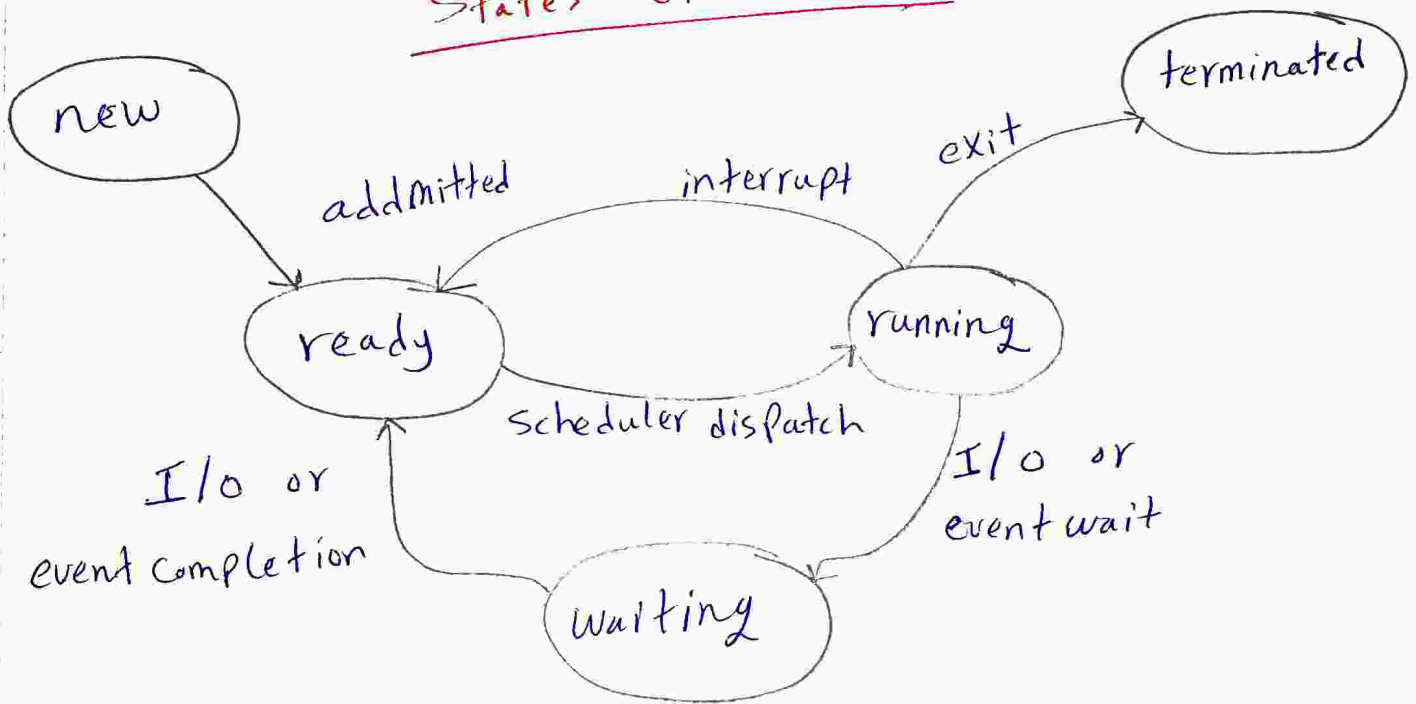
* Accounting

- the record of OS Keeping may be used for accounting or simply for accumulating uses statistics.

Process

- this word was first used by designers of Multics in 1960.
- used some what interchange ~~by~~ with task or job.
- Process means when a program in execution.

States of process



New → process is being created.

Running → instructions are being executed.

waiting → process is waiting for some event to occur.

Ready → process is waiting to be assigned to a processor.

Terminated → Process has finished executions.

→ As a process executed, it changes state.
The state of a process is defined in part by the current activity of process

Process Control Block

- Process are represented by PCB.
- Scheduler handles processes maintained in form of PCB.
- It acts as complete record to process & has the following parts.

Pointer	Process state
Process number	
Program Counter	
registers	
memory limits	
List of open files	
⋮	

"Process Control Block"

Process state

→ state may be new, ready, running, waiting and so on.

* Program Counter

→ PCB Contains the address of next instruction to be executed.

* CPU Registers

→ registers vary in number and type, depending on computer architecture.

→ They include accumulators, index register and stack pointer.

* Memory-management Information

→ PCB holds value of base & limit register, page table or depending on the memory system used by OS.

* I/O status information

→ information includes ~~the~~ list of I/O devices allocated to this process a list of open files etc allocated to the process is stored in PCB.

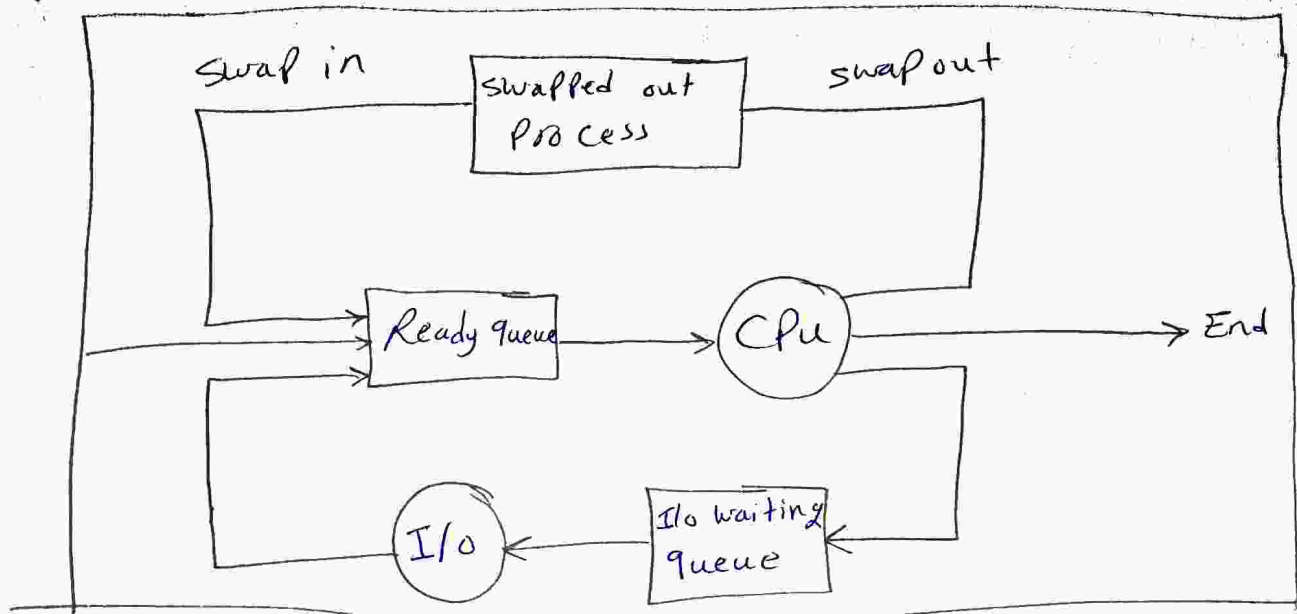
Scheduling

→ Fundamental operating system function.

→ every computer resource is scheduled before use.

→ is classified as

- 1) long term:- which process to be admit.
- 2) medium term:- which process to swap in or out.
- 3) short term:- which read process to execute next.



Inter Process - Communication (IPC)

→ means of information sharing and synchronization between process

→ In Computing, IPC is set of methods for the exchange of data among multiple threads in one or more process.

→ IPC in multiprocessor system, This allows a program to handle many user requests at the same time.

Threads

→ Processes that share an address space.

→ each thread has a Program Counter, register and run time stack.

→ There are multiple threads of control and single address space.

→ A threads sometimes called a lightweight process (LWP) is a basic unit of CPU utilization.

→ traditional or heavy weight process has a single thread of control.

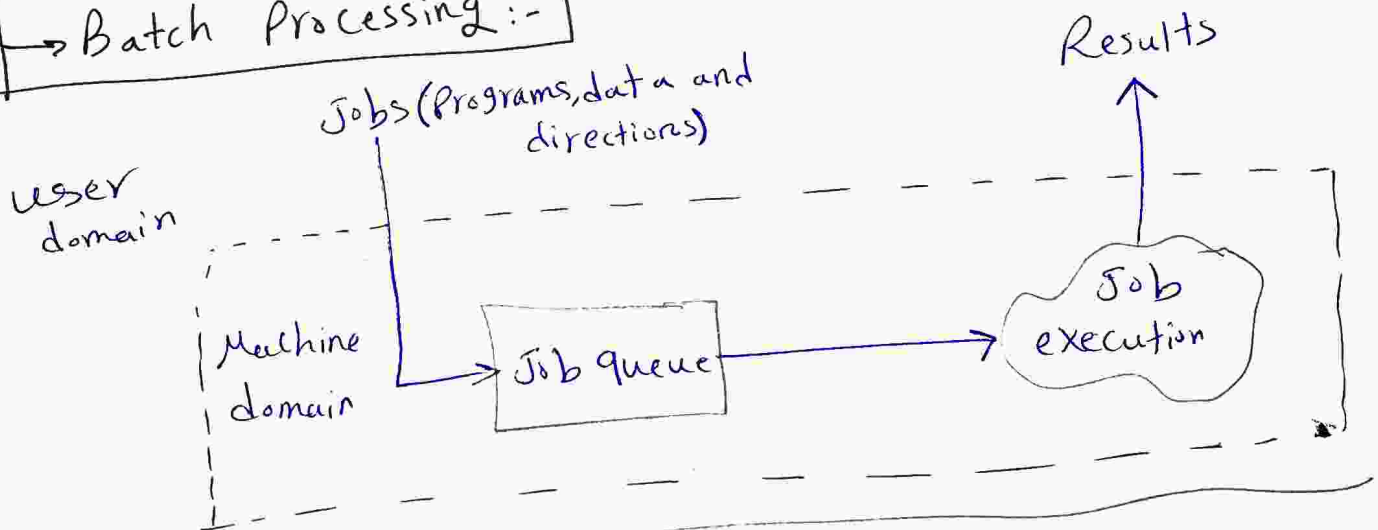
OS : Part 2

→ Functions of OS

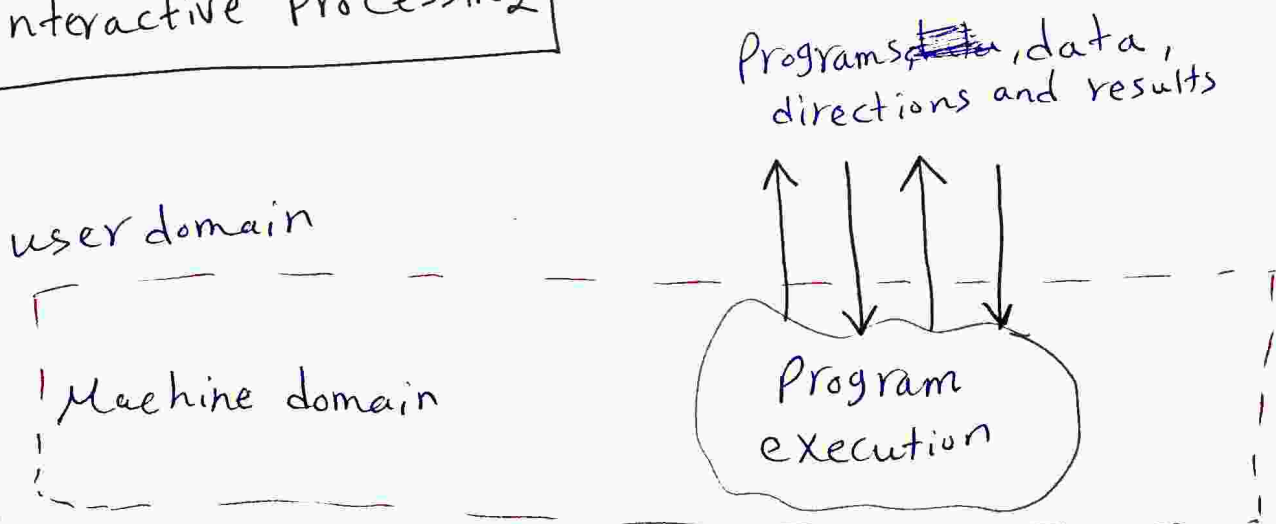
- 1) oversee operation of compiler.
- 2) store and retrieve files.
- 3) schedule programs for execution.
- 4) coordinate the execution of programs.

shared computation

→ Batch Processing :-



Interactive Processing



Time sharing / Multi Tasking

- 1) users are seeking services from same machine at the same time (time sharing)
- 2) Multiple terminals connected to same machine.
- 3) When multiprogramming is applied to single-user environments is called multitasking.

Multi Processor operating systems

- 1) Provide timesharing / multi-tasking capabilities by assigning different tasks to different processors as well as sharing the time on single processor.
- 2) Problems to solve
 - Load balancing: dynamically allocating tasks to the various processor so that all of them are used efficiently.
 - Scaling: breaking tasks into sub-tasks compatible with number of processors available.
- 3) Trend to develop a network wide "OS" rather than networks of individual "OS".

Embedded operating systems

- 1) used in hand held devices, mobile phones, cars, etc...
- 2) limited data storage and power ~~user~~ ^{conservation} are the big challenges.
- 3) Examples Vxworks, Palm OS, Rm Dos, Symbian.

Types of software

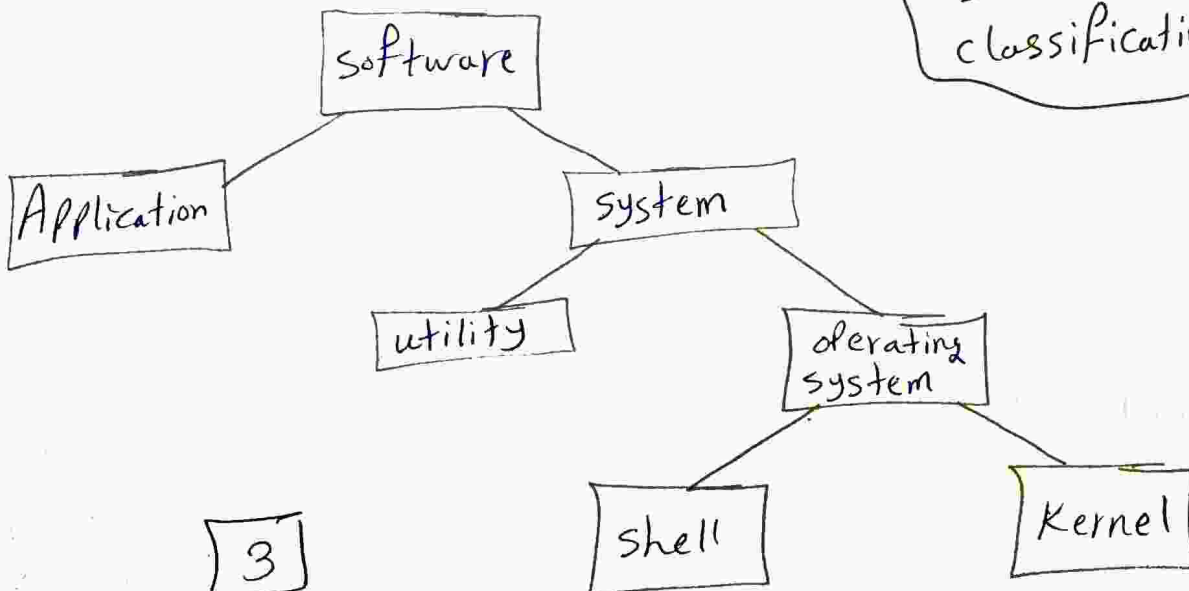
1) Application software

→ perform specific tasks for users: spread sheets, database systems, program development, games, ... etc

2) system software

→ provide infrastructure for application software

→ consists of operating system and utility software



software
classifications

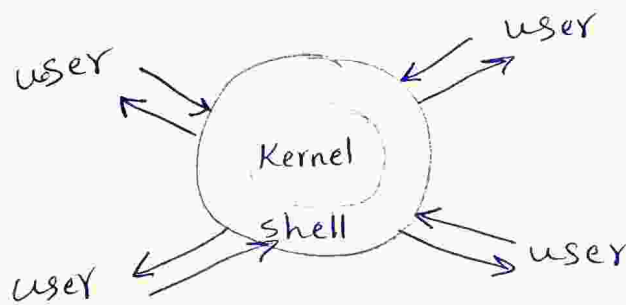
OS Components

*shell

- communicates with users. → text based
- Graphical user interface (GUI)

*Kernel : performs basic functions

- file manager → device drivers
 - memory manager. → process manager (scheduler, etc)
- ⇒ Shell is an interface between users and "OS".



*File manager

*Role ⇒ Coordinate the use of machine's mass storage facilities.

*Hierarchical organization

→ Directory : user-created bundle of files and other directories

→ Directory Path : sequence of directories within directories

* Access/operations to files is provided by File ~~manager~~ manager via a File descriptor

Device manager

- 1) Part of OS presented as collection of device drivers.
 - 2) Specialized software that communicate with the controllers to carry out operations on ~~the~~ peripheral devices connected to computer.
 - 3) each driver is specifically designed for its type of device.
-

Memory manager

- 1) Coordinating the use of main memory (allocate and deallocate space in main memory)
- 2) when total required memory space exceeds the physical available space.
→ may create illusion that machine has more memory than it actually does (virtual memory)

Process manager

* scheduler

→ Part of Kernel in charge with the strategy for allocation^{on} / de-allocation of CPU to each competing process

→ Add new processes to the process table and removes completed processes from process table.

* process table contains

- 1) memory area assigned to the process.
- 2) ~~priority~~ priority of the process.
- 3) state of process (ready or wait)

* Dispatcher

→ Component of Kernel that oversees execution of the scheduled processes.

→ Control allocation of CPU to the process in process table.

→ It performs "process switch" procedure to change ~~from~~ ^m from one process to another.

Process A → Dispatcher → Process B

Handling Competition for Resources

→ important task of OS is to allocate resources to the processes.

→ Semaphore → A control flag.

→ Mutual exclusion ⇒ Requirement for proper implementation of a critical region so that only one process at a time will execute sequence of instructions part of critical region

Deadlock

→ Another problem of resource Allocation:-

→ Processes block each other from continuing.

→ Conditions required for deadlock:

a) ~~Competition~~ Competition for non-sharable resources.

b) Resources requested on partial basis.

c) An Allocated resource can't be forcibly retrieved.

Security

- one of the role of OS is to provide security
- Attacks from outside
 - * Insecure Passwords
 - * Counter measures.
 - Auditing software.
- Attacks from within.
 - Counter measures → Control Process activities via Privileged (modes and instructions).

* Booting

- transfers OS from mass storage into the main memory.

* Bootstrap

- Program in ROM.
- Transfers OS from mass storage to main memory.
- Executes Jump to "OS".
- Run by CPU when power is turned on.

Lec 3: I/O devices

*input device :-

transforms information from the user into data that computer application can process.

- choice and method of use of input device should contribute positively to usability of the system.
- usability depends on the provision of appropriate feedback.

⇒ Pointing devices (cursor control)

a) 3D tracker

- Replaces mouse where desk space is limited (laptops)
- Relays position and orientation to a receiver.

b) Joystick

- Small stick, movable in any direction in a fixed socket.

c) Mouse

- Relative device. Buttons for discrete input.

d) Trackball

- Rotatable ball in fixed socket.

e) Tablet

- used with puck. Absolute device.

f) Mole → (foot mouse)

2) Data glove

↳ Communicates hand and finger positions to an app. used for manipulating virtual objects.

h) Touch-sensitive screen

↳ Special screen that detects the position of a finger touching it.

e) Light Pen

↳ location found by beam passing through screen during refresh cycle.

Device	Advantages	Disadvantages
Touch-screen	→ no moving parts. → durable low price.	→ tiring for prolonged use.
Light Pen	→ useful with flat screen → no training needed	→ not attractive by dark areas.
Mouse	fast, inexpensive, accurate.	→ requires flat, dedicated surface area.
Trackball	inexpensive, fine control, little desk space needed.	→ less easy to use than mouse.
Joystick	inexpensive, little desk space needed.	→ Can be inconvenient to use if built in

* Choosing appropriate input devices:-

- a) Matching devices with work.
 - natural mapping between use, feedback, meaning of result and user's mental model are needed.
 - Particular manipulations needed to accomplish a piece of work need to be analyzed.
- b) Matching devices with users.
- c) Matching devices with environment of use.
 - space. → relation to other concurrent tasks etc.,

Developments in input

⇒ Speech recognition

* advantages: minimal user training, freedom of hands opportunities for physically disabled.

* disadvantages: recognition system often needs training, liable to error; difficult for app. to interpret human speech.

⇒ Handwritten input

* advantages: easier to separate words, may need some training for users.

* disadvantages

→ Cursive script difficult to read, system may need training.

output devices

→ Provide information or feedback in a form which is understandable by humans.

* Visualisation

→ ~~Dynamic Visualisation~~

→ Dynamic Visualisation: became very important in information-rich applications.

→ The key issue is to find visual forms that support the users' mental model.

→ Perceptualisation → the multimedia equivalent of Visualisation.

→ 3D animation and virtual Reality are likely to increase in importance for "perceptual" interfaces.

Sound

→ It can complement a visual interface when users' attention is likely to turn away from a VDU screen.

→ important use → to deliver information on background events that need continual monitoring.

* "Seven Plus or minus two" rule for information overload applies to sounds, too.
→ Sound can be important in interfaces for the visually disabled.

Digital speech techniques

* Concatenation: involves:
a) digitally recording human speech in large chunks (word, short sentences)
b) reassembling it and play back.
e.g. → talking clock.

* Synthesis - by - rule involves synthesizing speech according to prescribes rules of sound formation to generate more "natural-sounding" tone.

Multimedia

→ Potential for multimedia includes the fact that it is estimated that only 7% of business critical information is "record-based".

→ multimedia databases including photos, video clips, sounds, etc., may permit digitalisation of much of the other 93%.

I/O Hardware

- Incredible variety of I/O devices.
- Common Concepts (Port-Bus-Controller)
- I/O instructions control devices.
- Devices have addresses used by
 - a) Direct I/O instructions.
 - b) Memory mapped I/O.

Polling

- Determine state of device
(Command ready - busy - Error)
- Busy - wait cycle ~~Port~~ to wait for I/O from device.

Interrupts

- interrupt handler receives interrupts.
- interrupt mechanism also used for ~~error~~ exceptions.
- ~~maskable to ignore or delay some~~ → maskable to ignore or delay some interrupts.
- interrupt vector to dispatch interrupt to correct handler.

interrupt-Driven I/O cycle ans ←

Page 7 Lec 3 Part 2 pdf

Direct memory Access

- used to avoid programmed I/O for large data movement.
- Requires DMA Controller.
- Bypasses CPU to transfer data directly between I/O device and memory.
- Six steps processes to perform DMA transfer. (on pdf)

Application I/O interface

Network devices

- Varying enough from block and character to have own interface.
- Approaches vary widely (Pipes, FIFO, Streams, mailboxes)

Clocks and Timers

- Provide current time, elapsed time, timer.
- ~~if~~ if Programmable interval time used for timing, periodic interrupts.
- ioctl (on UNIX) covers odd aspects of I/O such as clocks and timers.

Blocking and nonblocking I/O

* Blocking : Process suspended until I/O completed.

- Easy to use and understand.
- Insufficient for some needs.

* Non blocking

- I/O call returns as much as available.
- Implemented via multithreading.
- returns quickly with count of bytes ~~read~~ read or written.

* Asynchronous

- Process runs while I/O executes.
- Difficult to use.
- I/O ~~sub~~ subsystem signals process when I/O completed.

Kernel I/O Subsystem

* Scheduling

- Some OSs try fairness.
- Some I/O request ordering via per-device queue.

* Buffering store data in memory while transfer between devices.

- a) to maintain copy semantics.
- b) to cope with speed mismatch.
- c) to cope with device transfer size mismatch

* Caching

- Fast memory holding copy of data.
- Always just a copy.
- Key to performance.

* Spooling

- hold output for a device.
- if device can serve only one request at a time.
- For example: Printing.

* Device reservation

- Provide exclusive access to a device.
- system calls for allocation and deallocation.
- watch out for ~~death~~ deadlock.

Error handling

- OS can recover from disk read, device unavailable, transient write failures.
- most return an error number or code when I/O request fails.
- system error logs hold problem reports.

Kernel data structures

- Kernel keeps state info for I/O components, including open file tables, network connections.

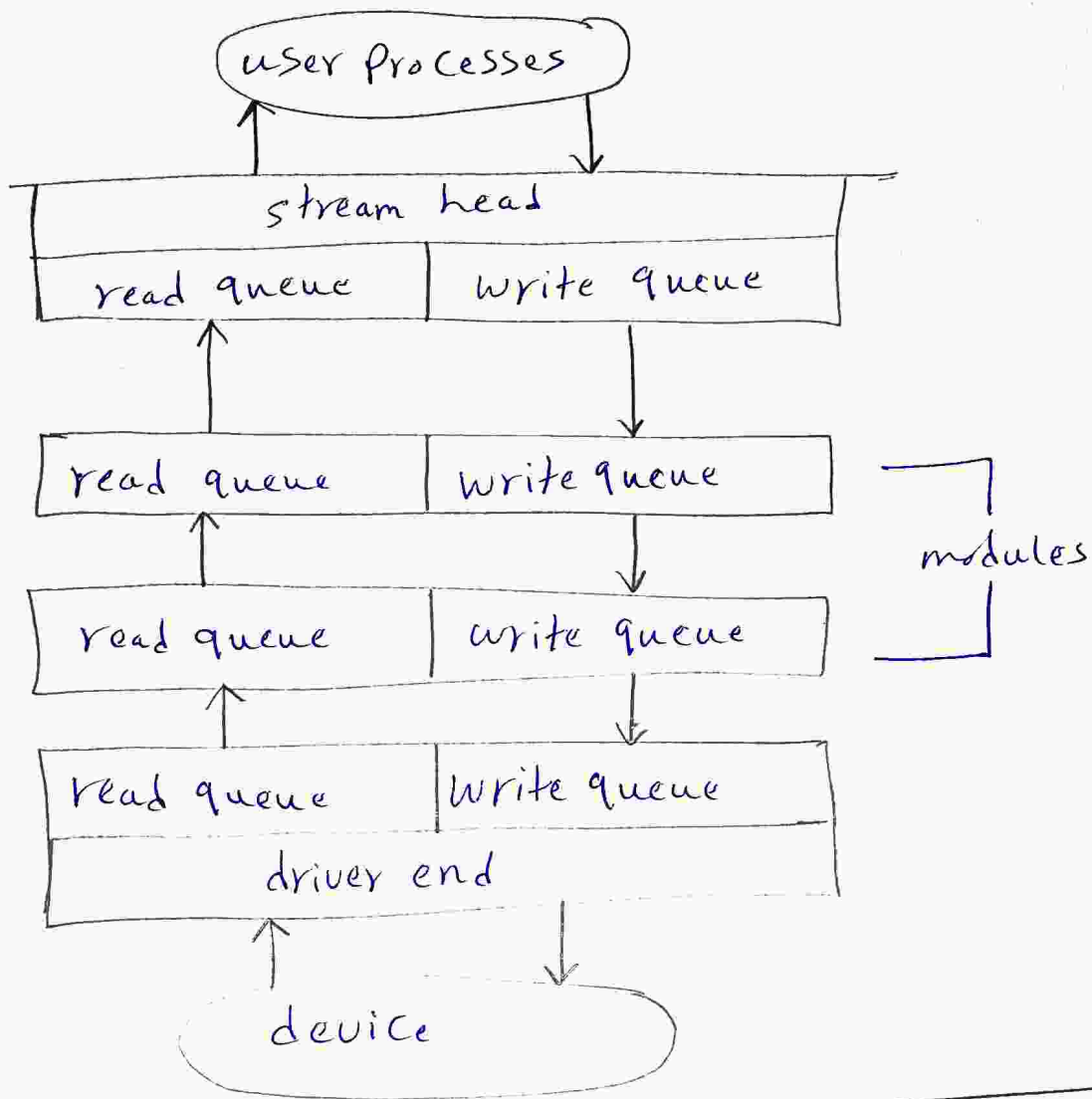
* Consider reading a file from disk for a process?

- Determine device holding file.
- Translate name to device representation.
- Physically read data from disk into buffer.
- make data available to requesting process.
- return control to process

Streams

- Full-duplex communication channel between user-level process and a device.
- Consists of:-
 - a) stream head interfaces with user process.
 - b) driver end interfaces with the device.
 - c) Zero or more stream modules between them.
- Each module contains read queue & write queue.
- message passing is used to communicate between queues.

stream structure



Performance

- I/O major factor in system performance:-
- Demands CPU to execute device driver, kernel I/O code.
 - Context switches due to interrupts.
 - Data copying.

Improving Performance

- 1) Reduce data copying. 2) use DMA.
 - 3) Reduce interrupt by using large transfers, smart controllers, polling.
 - 4) Balance CPU, memory, bus, and I/O Performance
~~to highest throughput.~~
-

← توجد مجموعة رسومات عجيبة، ومن غير
المعتقول! ستيعا بها بواسطة العقل البشري.

(12)

Lec 4

11

* Page management strategies:-

→ Fetch → Placement → replacement

Locality

→ Process tends to reference memory in highly localized patterns.

→ referenced Pages tend to ^{be} adjacent to one another in Process's virtual address space.

Fetch strategy

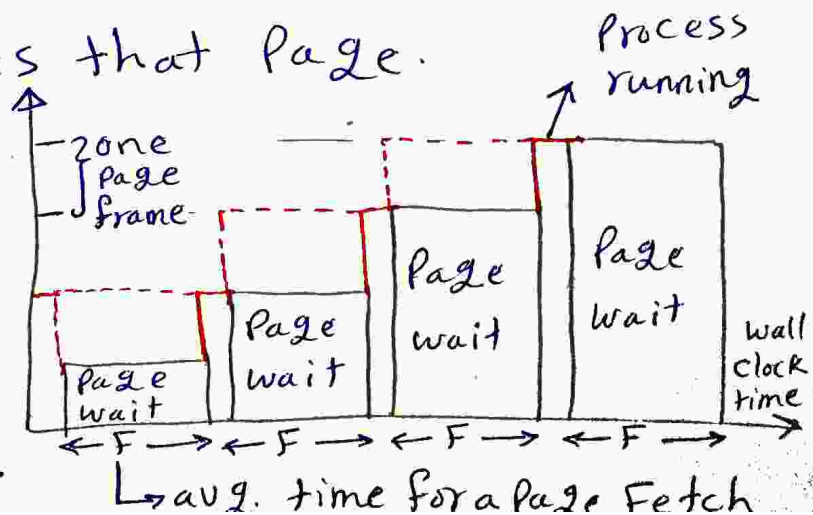
* 1) Demand Paging

→ When a Process first executes, the system loads ~~into main memory~~ the page that contains its first instruction into main memory.

→ After that, the system loads a page from secondary memory to main memory only when the process references that page.

→ requires a process to accumulate pages one at time

waiting process occupies memory



2) Anticipatory Paging

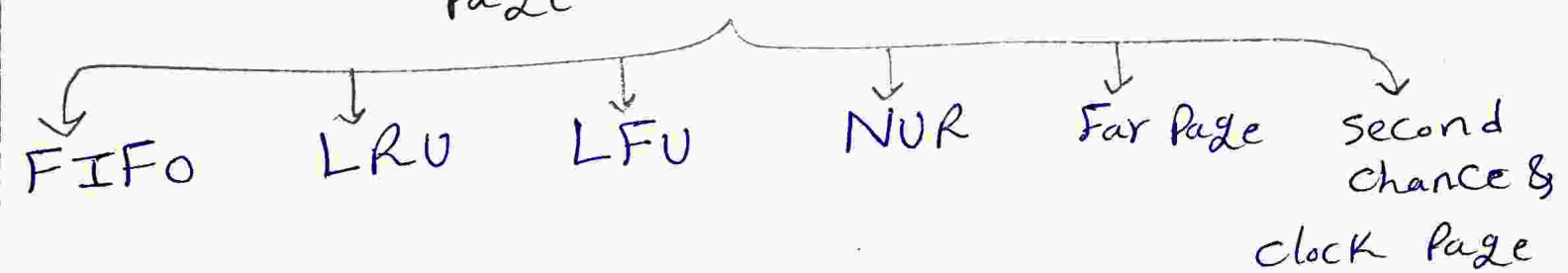
[2]

- It is attempt to Predict the Pages a Process will need and Preloads these Pages when memory is available.
- It must be carefully designed so that overhead incurred (النفقات المترتبة) by the strategy does not reduce system Performance.

Page replacement

- * ~~is~~ characterized by
 - heuristic it uses to select a Page for replacement
 - overhead it incurs.

Page replacement strategies



Random Page replacement

- low overhead.
- rarely used.
- each Page has an equal ~~probability~~ Probability.
- ~~not a difference~~ → not different with
- ~~not differ from~~ Particular Processes.

① First-In-First-out (FIFO) Page replacement

- replace heavily used pages.
- relatively low overhead. → simple queue.
- not practical for most systems.

② Least Recently used (LRU) Page replacement

⇒ Heuristic

- temporal locality.
- replace page that has not been recently used.

But

- increase system overhead.
- poor performance in certain situations

③ Last-Frequently-used (LFU) Page replacement

⇒ Heuristic

- Keep pages that are being used.
- replaces page that is least referenced.

But

- The page that referenced heavily in the past may never be referenced again. → implementation overhead.

4) Not-Used-Recently (NUR) Page replacement

4

⇒ Heuristic

- its goal to approximate LRU with less overhead.
- uses 2 indicator bits per page
 - referenced bit
 - modified bit
- bits are reset periodically.
- order for page replacement
 - un-referenced page
 - un-modified "
- NUR → supported in hardware on modern systems.

5) FIFO variation: second-chance Replacement

- examines referenced bit of the oldest page.
 - if off: page is replaced
 - if on: a) turns off the bit.
b) moves the page to tail of FIFO queue.
- Keep active pages in memory.

6) FIFO variation: clock page replacement

* Heuristic

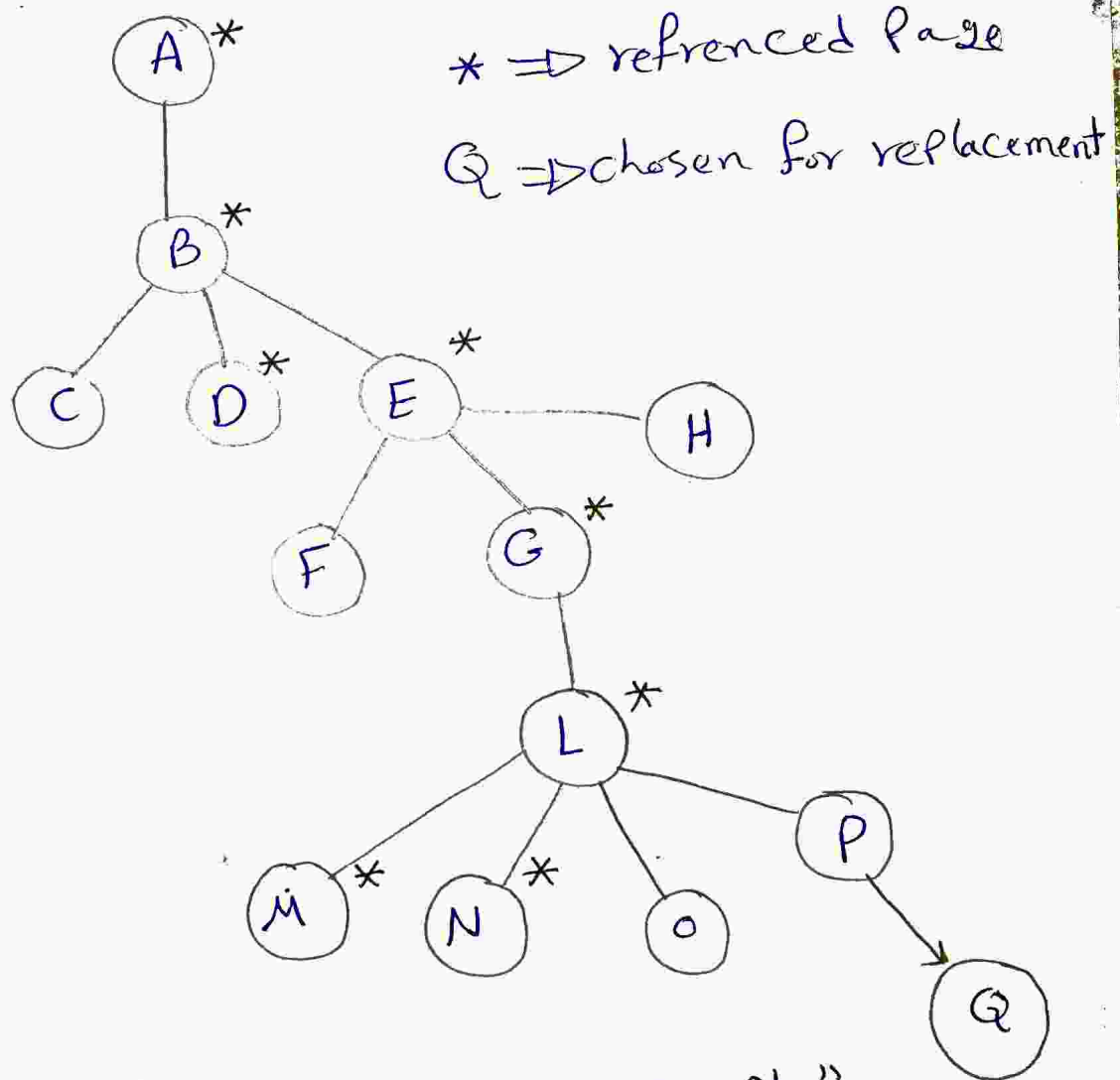
- uses circular list instead of FIFO queue.
- marker for oldest page.
- * Examines referenced bit of oldest page
 - if off ⇒ page is replaced.
 - if on ⇒ a) turn off the bit.
b) advances marker in circular list.

7) Far Page replacement

5

* Heuristic

- creates an access graph that characterizes a process's reference patterns.
- replace the unreferenced page that is far away from any referenced page in access graph.
- Performs at near-optimal levels.



"Far Page replacement : access graph"

Far Page replacement (Continued)

[6]

→ Performs at near-optimal levels.

But

- access graph need to be computed.
- access graph is complex to search and manage without hardware support.

Working set Model

→ to run a program efficiently: system must maintain that program's favored subset of pages in main memory.

* Heuristic

- Consider locality of page references.
- Keep "local" pages of process in memory.

* working-set-based page replacement strategy

→ Keep pages of working set in main memory.

But

- working set size changes.
- working set changes.
- overhead for working set management.

*Page-Fault-Frequency (PFF) Page Replacement

*Goal $\Rightarrow$ improve working set approach.

- $\rightarrow$ Adjusts a process's resident page set
- $\rightarrow$ Based on frequency at which process is faulting.
- $\rightarrow$ Based on time between page faults, called the process's interfault time.

PFF advantage

- $\rightarrow$ lower overhead
- $\rightarrow$ PFF adjusts resident page set only after each page fault.
- $\rightarrow$ working set management must run after each memory reference

Page release

* Problem

$\rightarrow$ inactive pages may remain in ~~memory~~ main memory.

* Solution

- $\rightarrow$ explicit voluntary page release.
- $\rightarrow$ need compiler and operating system support.

Page size

* Small Page sizes

- reduce internal fragmentation
- more memory available to other processes.
- can reduce the amount of memory required to contain a process's working set.

* Large Page sizes:

- reduce wasted memory from table fragmentation
- enable each TLB entry to map larger region of memory, improving performance.
- reduce number of I/O operations. ~~that~~

Global vs. local Page Replacement

- * Global: a) applied to all processes as a unit
- b) ignore characteristics of individual process behavior.

* Local

- consider each process individually.
- adjusts memory allocation.